NPS69-84-001

NAVAL POSTGRADUATE SCHOOL

Monterey, California

USE OF THE TENSOR PRODUCT FOR NUMERICAL WEATHER PREDICTION BY THE FINITE ELEMENT METHOD - PART 1

R. E. NEWTON

April 1984

Final Report for Period
October 1982 - September 1983

Approved for Public Release; Distribution Unlimited

...ared for:

...l Environmental Prediction Research Facility
...erey, California 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Commodore R. H. Shumaker
Superintendent

D. A. Schrady
Provost

The work reported herein was supported by the Naval Environmental Prediction Research Facility.

This report was prepared by:

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS69-84-001	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) USE OF THE TENSOR PRODUCT FOR NUMERICAL WEATHER PREDICTION BY THE FINITE ELEMENT METHOD - PART 1.		5. TYPE OF REPORT & PERIOD COVERED Final, October 82 - September 83
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. E. Newton		8. CONTRACT OR GRANT NUMBER(s) N6685684WR84103
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Environmental Prediction Research Facility Monterey, California 93943		12. REPORT DATE April 1984
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite element, numerical weather prediction, tensor product

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report concerns application of the tensor product in solving large sets of linear equations arising in Numerical Weather Prediction problems. The coefficient matrix considered is called the "mass" matrix in finite element parlance. Theory of the tensor product resolution is presented, along with methods for imposing Dirichlet and cyclic boundary conditions. Operation

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT

counts and storage requirements are compared with corresponding numbers for some widely-used solution algorithms. Listings of Fortran programs for implementing the tensor product solution system (TENSOR) and testing it are given.

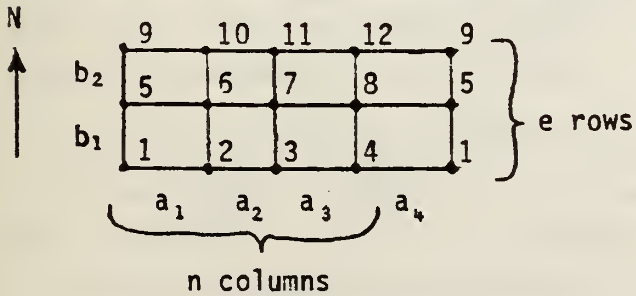
USE OF THE TENSOR PRODUCT FOR NUMERICAL WEATHER PREDICTION BY THE FINITE ELEMENT METHOD - PART 1.

Introduction

In Ref. 1 Hinsman has developed a Finite Element (FE) program for Numerical Weather Prediction applications. The grid employed is rectangular with nodes at the intersections of north-south and east-west lines. It was shown by Staniforth and Mitchell (Ref. 2) that the coefficient matrices for such a grid could be expressed as tensor products. In these products the factors are matrices which depend solely on grid spacing in the two orthogonal directions. This report deals with the coefficient matrix called the "mass" matrix in FE parlance. (In Refs. 3 and 4 applications of the tensor product resolution to the FE "stiffness" matrix are considered.) The theory which underlies the economical computational scheme based on the mass matrix resolution is first presented. Next, the number of floating point operations and the number of storage locations needed for the coefficient matrix of this scheme are compared with those required by other better-known algorithms. A set of FORTRAN subroutines for implementing the tensor product scheme (TENSOR) is given in Appendix B.

Theory

Consider the grid shown in Fig. 1. There are n spaces



(Note the cyclic boundary condition in the E-W direction.)

Fig. 1. Node numbering and spacing.

along each of e grid lines in the east-west direction. Node numbering is from west to east along successive grid lines,

beginning in the southwest corner. There is a cyclic boundary condition in the east-west direction so that the node number appearing at the beginning of each horizontal row is repeated at the end. Spacings of the horizontal and the vertical grid lines are not necessarily uniform.

The computational problem addressed here is the solution of the equation

$$Mw = v \qquad \langle 1\rangle$$

where M (the "mass" matrix) is a square, symmetric matrix of size ne and w and v are column vectors of height ne. M and v are input quantities and w is sought. The tensor product representation of M is

$$M = MB * MA \qquad \langle 2\rangle$$

where MB is a square, symmetric, tridiagonal matrix of size e and MA is also square, symmetric and of size n. MA is tridiagonal except for nonzero elements in upper right and lower left corners. MB depends solely on the north-south node spacing b and MA depends upon the east-west node spacing a . The asterisk (*) denotes the tensor product. Explicit expressions for matrices MA, MB, and the tensor product are given in Appendix A.

Let MB be represented as (e = 3)

$$MB = \begin{bmatrix} mb_{11} & mb_{12} & 0 \\ mb_{21} & mb_{22} & mb_{23} \\ 0 & mb_{32} & mb_{33} \end{bmatrix} \qquad \langle 3\rangle$$

If we partition w and v into e n x 1 subvectors so that

$$w = \begin{bmatrix} w_I \\ w_{II} \\ w_{III} \end{bmatrix} \qquad v = \begin{bmatrix} v_I \\ v_{II} \\ v_{III} \end{bmatrix} \qquad \langle 4\rangle$$

we may use <2>, <3> and <4> to rewrite <1> as

$$\begin{aligned} mb_{11}\, MA\, w_I + mb_{12}\, MA\, w_{II} \phantom{+ mb_{23}\, MA\, w_{III}} &= v_I \\ mb_{21}\, MA\, w_I + mb_{22}\, MA\, w_{II} + mb_{23}\, MA\, w_{III} &= v_{II} \qquad \langle 5\rangle \\ mb_{32}\, MA\, w_{II} + mb_{33}\, MA\, w_{III} &= v_{III} \end{aligned}$$

Define $$W = \langle w_I \; w_{II} \; w_{III} \rangle$$

<6>

and $$V = \langle v_I \; v_{II} \; v_{III} \rangle$$

It is easy to verify that the equations <5> are equivalent to

$$MA \; W \; MB = V \qquad \langle 7 \rangle$$

Solution of <7> can be accomplished by standard Gaussian elimination procedures. Specifically, the following steps are required.

(1) LDLT factoring of MA (n x n).
(2) Forward reduction and back-substitution for e right-hand side vectors.
(3) LDLT factoring of MB (e x e).
(4) Forward reduction and back-substitution for n right-hand side vectors.

This entire process is economical of both storage and arithmetic operations because of the tridiagonal structure of MA and MB.

Boundary Conditions

The cyclic boundary condition is implemented by repeating the node numbers of the western boundary on the eastern boundary as shown in Fig. 1. As already noted, this accounts for nonzero entries in the upper right-hand and lower left-hand corners of MA.

It is sometimes required to impose a Dirichlet boundary condition on the southern and northern boundaries of the region. Specifically, the subvectors w and w of the solution vector w are prescribed. To implement this boundary condition the following modifications to the standard solution procedure are required.

In the n x e matrix V on the right-hand side of <7> the first and last columns are replaced by the prescribed boundary values of w, i.e., put $v_I = w_I$ and $v_e = w_e$. Let $X = W \; MB$ and solve the system

$$MA \; X = V \qquad \langle 8 \rangle$$

processing successive columns of V in standard fashion, but omitting the first and last columns. The reduced problem

now takes the form W MB = X and the first and last columns of X are w_I and w_e, respectively. Transposing both sides of this equation gives

MB WT = XT <9>

where WT and XT are the respective transposes of W and X (recall that MB is symmetric and is thus not altered on transposition). Since the first and last rows of WT are known, the corresponding scalar equations are not needed. Accordingly, we form MB1 by deleting the first and last rows of MB. We also reduce XT to XT1 by omitting the first and last rows. This leaves the result

MB1 WT = XT1 <10>

or, in extenso, this takes the form (for n = 3, e = 5)

$$\begin{bmatrix} mb_{21} & mb_{22} & mb_{23} & 0 & 0 \\ 0 & mb_{32} & mb_{33} & mb_{34} & 0 \\ 0 & 0 & mb_{43} & mb_{44} & mb_{45} \end{bmatrix} \begin{bmatrix} k & k & k \\ u & u & u \\ u & u & u \\ u & u & u \\ k & k & k \end{bmatrix} = \begin{bmatrix} k & k & k \\ k & k & k \\ k & k & k \end{bmatrix}$$

(In WT and XT1 the elements denoted by "k" are known and those denoted by "u" are unknown.) This equation may be put in standard form by first altering the first row of XT1 by subtracting mb_{21} times the corresponding entries in the first row of WT and altering the last row of XT1 by subtracting mb_{45} times the corresponding entries in the last row of WT. Calling the new right hand side XT2 and forming MB2 from MB1 by discarding the first and last columns and forming WT1 by discarding the first and last rows of WT, the result is

MB2 WT1 = XT2 <11>

Solution of <11> is carried out by LDLT factoring of MB2, followed by forward reduction and back substitution.

Floating Point Operations and Matrix Storage Requirements

Presented here is a comparison of floating point operation counts and matrix storage requirements for the tensor product scheme and three widely-used solution algorithms for

solving <7> or its equivalent <1>. Three of the schemes take advantage of symmetry of the coefficient matrices and store only elements on or above the principal diagonal. One of these, the "band solver" (BAND), places these elements in a rectangular matrix ne x r, where r is the maximum row length of the upper triangle of M. The "sky-line solver" (SKY) further economizes by storing only that part of the upper triangle beginning at the diagonal and extending to the topmost nonzero element of each column. These subvectors are assembled into a single vector. This scheme requires an additional integer address vector of length ne + 1. The remaining algorithm, "successive over-relaxation" (SOR), is iterative rather than direct.

In most applications of the direct solvers the number of floating point operations required to factor the coefficient matrix into LDLT form is much greater than those required to complete the process of finding a single solution vector w corresponding to a given right-hand side vector v (forward reduction and back-substitution). In the present application, however, the latter solution process must be carried out 17 times for each time step, so that the LDLT factoring makes a negligible contribution to the total computational expenditure. Accordingly, the operations required for factoring are not included in the tabulation below.

In the following table the results given for the number of floating point operations are given in terms of the grid parameters n and e (defined in Fig. 1). One multiplication (or one division) plus one addition (or one subtraction) is counted as one operation. Exact results for these operation counts would take the form of a polynomial in n and e. Only the highest degree terms are given in the table. Since it is not possible to predict the number of iterations per solution when using SOR, the operation count given for that algorithm is for a single <u>iteration</u>. Also, since the number of storage locations required for SOR coefficient matrices is highly grid-dependent, no such entry is given for SOR.

TABLE I. Operation Counts and Storage Requirements.

Algorithm	Number of Operations per Solution	Number of Storage Locations for Coefficient Matrices
SOR	10 en (1)	--- (2)
SKY	2 en^2	en^2
BAND	4 en^2	2 en^2
TENSOR	8 en	3 n + 4 e

Notes: 1. Number of operations per iteration.
2. Number of storage locations is grid-dependent.

Conclusion

Close comparison of operation counts and storage requirements leads to the conclusion that the TENSOR algorithm is clearly superior to the SKY and BAND algorithms. The comparison with SOR is not as clear-cut. Considering, however, that the operation count for SOR is for only one iteration, there really seems to be little doubt that TENSOR is the method of choice.

<u>List of References</u>

1. Hinsman, D. E., "Numerical Simulation of Atmospheric Flow on Variable Grids using the Galerkin Finite Element Method," Doctoral Dissertation, Naval Postgraduate School, March 1983.

2. Staniforth, A. N., and H. L. Mitchell, "A Semi-Implicit Finite Element Barotropic Model," Monthly Weather Review, v. 105, p. 154-169, February 1977.

3. Dorr, F. W., "The Direct Solution of the Discrete Poisson Equation on a Rectangle," SIAM Review, v. 29, p. 248-263, April 1970.

4. Lynch, R. E., J. R. Rice and D. H. Thomas, "Tensor Product Analysis of Partial Difference Equations," Bull. Amer. Math. Soc. v. 70, p. 378-384, 1964.

5. Bathe, K. J., <u>Finite Element Procedures in Engineering Analysis</u>, p. 721,722, Prentice-Hall, 1982.

APPENDIX A
MATRICES MA, MB, AND THE TENSOR PRODUCT

Symbols a_i and b_i which appear in MA and MB are defined in Fig. 1.

$$MA = \frac{1}{6}\begin{bmatrix} 2(a_4+a_1) & a_1 & 0 & a_4 \\ a_1 & 2(a_1+a_2) & a_2 & 0 \\ 0 & a_2 & 2(a_2+a_3) & a_3 \\ a_4 & 0 & a_3 & 2(a_3+a_4) \end{bmatrix}$$
$$(n = 4)$$

$$MB = \frac{1}{6}\begin{bmatrix} 2b_1 & b_1 & 0 \\ b_1 & 2(b_1+b_2) & b_2 \\ 0 & b_2 & 2b_2 \end{bmatrix}$$
$$(e = 3)$$

The tensor product of matrices C and D may be represented in block partition form as

$$C*D = \begin{bmatrix} c_{11}D & c_{12}D & c_{13}D \\ c_{21}D & c_{22}D & c_{23}D \\ c_{31}D & c_{32}D & c_{33}D \end{bmatrix}$$

where the c_{ij} are the elements of C. Note that, if C and D have dimensions r x s and t x u, respectively, the tensor product has dimensions rt x su.

APPENDIX B

FORTRAN PROGRAM LISTINGS

FORTRAN programs for the implementation of TENSOR are listed here. They appear in the form of subroutines AMTRX2, FACTOR, BACKA, and BACKB within the test program GAUSS3. (The subroutines FACTOR, BACKA, and BACKB are adapted from subroutine COLSOL of Ref. 5.) Also included are GOG3, an Exec used to execute GAUSS2 - a dimensioned version of GAUSS3, and CDIM, an Xedit program used to enter the dimensions.

Listing: GAUSS3 FORTRAN

```
C     MAIN PROGRAM    MASS MATRIX USING TENSOR PRODUCT FACTORS
C
C     THIS PROGRAM IS DESIGNED TO TEST THE SCHEME (TENSOR)
C     WHICH RESOLVES THE MASS MATRIX INTO A TENSOR PRODUCT IN
C     ORDER TO SOLVE THE SYSTEM OF EQUATIONS M w = v.  THE
C     SUBROUTINES MAY BE INSERTED IN THE PROGRAM DEVISED BY
C     HINSMAN.
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/CM1A/NLAT,NLONG
      COMMON/CM8/A(Z1),B(Z1)
      COMMON AG(ZB),BG(ZC),GA(ZK),GB1(ZL),GB2(ZL),MA(ZM),
     1MB(ZN)
      DIMENSION V(ZP)
      READ(5,*)NLONG, NLAT
      LATX=NLAT+1
      WRITE(6,1000)
1000  FORMAT(//,'    MASS MATRIX - TENSOR PRODUCT RESOLUTION'
     *,/)
      WRITE(6,1001)NLONG,NLAT
      READ(5,*)A,B
      WRITE(6,500)A
      WRITE(6,503)B
503   FORMAT(/,'   B:  ',(24F3.0))
500   FORMAT(/,'   A:  ',(24F3.0))
1001  FORMAT('    NLONG =',I3,'       NLAT =',I3,/)
C   CONSTRUCT FACTORS, GA, GB1, AND GB2, OF MASS MATRIX
      CALL AMTRX2
      WRITE(6,501)AG
501   FORMAT(/,'    AG:   ',(12F4.1))
      WRITE(6,504)BG
504   FORMAT(/,'    BG:   ',(12F4.1))
      WRITE(6,1002)GA
1002  FORMAT(/,'    GA',/,(3X,6F7.3))
      WRITE(6,1004)GB1
1004  FORMAT(//,'    GB1',/,(3X,6F7.3))
      WRITE(6,1005)GB2
1005  FORMAT(//,'    GB2',/,(3X,6F7.3))
      WRITE(6,1003)MA
      WRITE(6,1006)MB
      CU=GB1(3)
      CL=GB1(2*LATX-1)
      K=(NLAT-1)*NLONG
C   IF NDIR>0, THERE IS A DIRICHLET BOUNDARY CONDITION ON
C   NORTHERN AND SOUTHERN BOUNDARIES.
      READ(5,*)NDIR,V
      WRITE(6,502)NDIR,V
502   FORMAT(/,' NDIR = ',I1,/,' V: ',6F8.2,/,(4X,6F8.2))
C   PERFORM LDLT FACTORING OF GA, GB1, AND GB2
      CALL FACTOR(GA,MA,NLONG)
      CALL FACTOR(GB1,MB,LATX)
      CALL FACTOR(GB2,MB,LATX)
      WRITE(6,1002)GA
      WRITE(6,1004)GB1
      WRITE(6,1005)GB2
C   PERFORM FORWARD REDUCTION AND BACK-SUBSTITUTION USING
C   FACTORS OF GA
      CALL BACKA(GA,V,MA,NDIR)
C   DIRICHLET BOUNDARY CONDITION ON NORTH AND SOUTH
C   BOUNDARIES ?
      IF(NDIR.GT.0)GO TO 3
      WRITE(6,510)V
C   PERFORM FORWARD REDUCTION AND BACK-SUBSTITUTION USING
C   FACTORS OF GB1
      CALL BACKB(GB1,V,MB,NDIR)
      GO TO 6
C   CORRECT RIGHT-HAND SIDE FOR DIRICHLET CONDITION
3     DO 2 J=1,NLONG
      V(J+NLONG)=V(J+NLONG)-CU*V(J)
2     V(J+K)=V(J+K)-CL*V(J+NLAT*NLONG)
      WRITE(6,510)V
C   PERFORM FORWARD REDUCTION AND BACK-SUBSTITUTION USING
C   FACTORS OF GB2
      CALL BACKB(GB2,V,MB,NDIR)
```

```
6     WRITE(6,510)V
510   FORMAT(/,' V: ',6F8.2,/,(4X,6F8.2))
C  READ A NEW RIGHT-HAND SIDE AND PERFORM A SECOND SOLUTION
      READ(5,*)NDIR,V
      WRITE(6,502)NDIR,V
      CALL BACKA(GA,V,MA,NDIR)
      IF(NDIR.GT.0)GO TO 7
      WRITE(6,510)V
      CALL BACKB(GB1,V,MB,NDIR)
      GO TO 16
7     DO 5 J=1,NLONG
      V(J+NLONG)=V(J+NLONG)-CU*V(J)
5     V(J+K)=V(J+K)-CL*V(J+NLAT*NLONG)
      WRITE(6,510)V
      CALL BACKB(GB2,V,MB,NDIR)
16    WRITE(6,510)V
1003  FORMAT(/,'   MA:',2X,36I3)
1006  FORMAT(/,'   MB:',2X,36I3)
      STOP
      END
C *************************************************************
      SUBROUTINE FACTOR(A,MAXA,NN)
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .   - - INPUT VARIABLES - -                                    .
C . A(NWK)      = STIFFNESS MATRIX STORED IN COMPACTED FORM     .
C . MAXA(NNP) = VECTOR CONTAINING ADDRESSES OF DIAGONAL         .
C .               ELEMENTS OF STIFFNESS MATRIX IN A             .
C . NN          = NUMBER OF EQUATIONS                            .
C . NWK         = NUMBER OF ELEMENTS BELOW SKYLINE               .
C .   - - OUTPUT - -                                             .
C . A(NWK)      = D AND L - FACTORS OF STIFFNESS MATRIX         .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),MAXA(1)
C
C     PERFORM L*D*LT FACTORIZATION OF STIFFNESS MATRIX
C
40    DO 140 N=1,NN
      KN=MAXA(N)
      KL=KN+1
      KU=MAXA(N+1)-1
      KH=KU-KL
      IF(KH)110,90,50
50    K=N-KH
      IC=0
      KLT=KU
      DO 80 J=1,KH
      IC=IC+1
      KLT=KLT-1
      KI=MAXA(K)
      ND=MAXA(K+1)-KI-1
      IF(ND)80,80,60
60    KK=MIN0(IC,ND)
      C=0.
      DO 70 L=1,KK
70    C=C+A(KI+L)*A(KLT+L)
      A(KLT)=A(KLT)-C
80    K=K+1
90    K=N
      B=0.
      DO 100 KK=KL,KU
      K=K-1
      KI=MAXA(K)
      C=A(KK)/A(KI)
      B=B+C*A(KK)
100   A(KK)=C
      A(KN)=A(KN)-B
110   IF(A(KN))120,120,140
120   WRITE(IOUT,2000)N,A(KN)
      STOP
140   CONTINUE
2000  FORMAT(//,' STOP - STIFFNESS MATRIX NOT POSITIVE
     1DEFINITE',//,' NONPOSITIVE PIVOT FOR EQUATION
     2',I4,//,' PIVOT = ',E20.12)
      RETURN
      END
```

```
C ***********************************************************************
      SUBROUTINE BACKA(A,V,MAXA,NDIR)
C ***********************************************************************
C  THIS SUBROUTINE PERFORMS THE FORWARD REDUCTION AND BACK-
C  SUBSTITUTION USING THE FACTORS OF GA
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/CM1A/NLAT,NLONG
      DIMENSION A(1),V(1),MAXA(1)
C
C     REDUCE RIGHT-HAND-SIDE LOAD VECTOR
C
      JMIN=1
      LATX=NLAT+1
      JMAX=LATX
C  IS THERE A DIRICHLET BOUNDARY CONDITION?
      IF(NDIR.LT.1)GO TO 140
C  SKIP NORTH AND SOUTH BOUNDARIES
      JMIN=2
      JMAX=NLAT
140   DO 240 J=JMIN,JMAX
150   DO 180 N=1,NLONG
      KL=MAXA(N)+1
      KU=MAXA(N+1)-1
      IF(KU-KL)180,160,160
160   K=N
      C=0.
      DO 170 KK=KL,KU
      K=K-1
170   C=C+A(KK)*V(K+(J-1)*NLONG)
      V(N+(J-1)*NLONG)=V(N+(J-1)*NLONG)-C
180   CONTINUE
C
C     BACK-SUBSTITUTE
C
      DO 200 N=1,NLONG
      K=MAXA(N)
200   V(N+(J-1)*NLONG)=V(N+(J-1)*NLONG)/A(K)
      N=NLONG
      DO 230 L=2,NLONG
      KL=MAXA(N)+1
      KU=MAXA(N+1)-1
      IF(KU-KL)230,210,210
210   K=N
      DO 220 KK=KL,KU
      K=K-1
220   V(K+(J-1)*NLONG)=V(K+(J-1)*NLONG)-A(KK)*V(N+(J-1)*
     1NLONG)
230   N=N-1
240   CONTINUE
      RETURN
      END
C
C ***********************************************************************
      SUBROUTINE BACKB(A,V,MAXA,NDIR)
C ***********************************************************************
C  THIS SUBROUTINE PERFORMS THE FORWARD REDUCTION AND BACK-
C  SUBSTITUTION USING THE FACTORS OF GB1 OR GB2
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/CM1A/NLAT,NLONG
      DIMENSION A(1),V(1),MAXA(1)
C
C     REDUCE RIGHT-HAND-SIDE LOAD VECTOR
C
      LATX=NLAT+1
      NMIN=1
      NMAX=LATX
C  IS THERE A DIRICHLET BOUNDARY CONDITION?
      IF(NDIR.LT.1)GO TO 50
C  SKIP NORTH AND SOUTH BOUNDARIES
      NMIN=2
      NMAX=NLAT
50    DO 240 J=1,NLONG
150   DO 180 N=NMIN,NMAX
      KL=MAXA(N)+1
```

```
      KU=MAXA(N+1)-1
      IF(KU-KL)180,160,160
160   K=N
      C=0.
      DO 170 KK=KL,KU
      K=K-1
170   C=C+A(KK)*V(J+(K-1)*NLONG)
      V(J+(N-1)*NLONG)=V(J+(N-1)*NLONG)-C
180   CONTINUE
C
C     BACK-SUBSTITUTE
C
      DO 200 N=NMIN,NMAX
      K=MAXA(N)
200   V(J+(N-1)*NLONG)=V(J+(N-1)*NLONG)/A(K)
      LMIN =2
      LMAX=LATX
      IF(NDIR.LT.1)GO TO 205
      LMIN=3
      LMAX=NLAT
205   N=LMAX
      DO 230 L=LMIN,LMAX
      KL=MAXA(N)+1
      KU=MAXA(N+1)-1
      IF(KU-KL)230,210,210
210   K=N
      DO 220 KK=KL,KU
      K=K-1
220   V(J+(K-1)*NLONG)=V(J+(K-1)*NLONG)-A(KK)*V(J+(N-1)*
     1NLONG)
230   N=N-1
240   CONTINUE
      RETURN
      END
C
C ***************************************************************
      SUBROUTINE AMTRX2
C ***************************************************************
C   THIS SUBROUTINE FORMS THE MASS MATRIX IN THE FORM OF A
C   TENSOR PRODUCT OF THE GB MATRIX AND THE GA MATRIX.
C   THE FIRST OF THESE IS NLAT + 1 BY NLAT + 1,SYMMETRIC,
C   AND TRIDIAGONAL.  THE SECOND IS NLONG BY NLONG,
C   SYMMETRIC, AND TRIDIAGONAL EXCEPT FOR SINGLE ELEMENTS
C   IN UPPER RIGHT HAND AND LOWER LEFT HAND CORNERS. GB IS
C   STORED IN SKYLINE VECTOR FORM (UPPER TRIANGLE WITH SPACE
C   FOR FILL-IN) AS GB1 AND GB2.  THE LATTER VERSION IMPOSES
C   A DIRICHLET BOUNDARY CONDITION ON THE NORTH AND SOUTH
C   BOUNDARIES.  GA IS ALSO STORED IN SKYLINE VECTOR FORM.
C   INTEGER ADDRESS VECTORS MB AND MA ARE ALSO GENERATED.
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/CM1A/NLAT,NLONG
      COMMON/CM8/A(Z1),B(Z1)
      COMMON AG(ZB),BG(ZC),GA(ZK),GB1(ZL),GB2(ZL),MA(ZM),
     1MB(ZN)
C   DIMENSION BG(NLAT),AG(NLONG),GB1(2*NLAT-1),
C   GA(3*NLONG-3),MA(NLONG+1),MB(NLAT+2)
      LATX=NLAT+1
C   FIND BG = (ELEMENT HEIGHT)/6.
      DO 2 J=1,NLAT
2     BG(J)=B(1+NLONG*(J-1))/3.
C   GENERATE GB1 AND GB2
      GB1(1)=2.*BG(1)
      GB2(1)=1.
      DO 4 J=2,NLAT
      K=2*(J-1)
      GB1(K)=2.*(BG(J-1)+BG(J))
      GB1(K+1)=BG(J-1)
      GB2(K)=GB1(K)
4     GB2(K+1)=GB1(K+1)
      GB1(2*NLAT)=2.*BG(NLAT)
      GB1(2*NLAT+1)=BG(NLAT)
      GB2(3)=0.
      GB2(2*NLAT)=1.
      GB2(2*NLAT+1)=0.
C   FIND AG = (ELEMENT WIDTH)/6.
```

```
      DO 10 J=1,NLONG
10    AG(J)=A(J)/3.
C   GENERATE GA
      GA(1)=2.*(AG(1)+AG(NLONG))
      DO 12 J=2,NLONG
      K=2*(J-1)
      GA(K)=2.*(AG(J-1)+AG(J))
12    GA(K+1)=AG(J-1)
      K1=2*NLONG
      K2=3*NLONG-4
      DO 14 K=K1,K2
14    GA(K)=0.
      GA(3*NLONG-3)=AG(NLONG)
C   GENERATE DIRECTORY VECTORS
      MB(1)=1
      DO16 J=1,NLAT
16    MB(J+1)=2*J
      MB(NLAT+2)=2*(NLAT+1)
      MA(1)=1
      DO 18 J=2,NLONG
18    MA(J)=2*(J-1)
      MA(NLONG+1)=3*NLONG-2
      RETURN
      END
```

Listing: GOG3 EXEC

```
ERASE GAUSS2 * A1
COPY GAUSS3 FORTRAN A1 GAUSS2 = =
&STACK CDIM
&STACK FILE
X GAUSS2 FORTRAN
FORTGI GAUSS2
GLOBAL TXTLIB FORTMOD2 MOD2EEH
FILEDEF 05 DISK
LOAD GAUSS2 (START
```

Listing: CDIM XEDIT

```
SET CMSTYPE HT
TOP
* ZB=NLONG
C /ZB/6/ * *
TOP
* ZC=NLAT
C /ZC/3/ * *
TOP
* Z1=NLONG*NLAT
C /Z1/18/ * *
TOP
* ZK=3*NLONG-3
C /ZK/15/ * *
TOP
ZL=2*NLAT+1
C /ZL/7/ * *
TOP
* ZM=NLONG+1
C /ZM/7/ * *
TOP
* ZN=NLAT+2
C /ZN/5/ * *
TOP
* ZP=NLONG*(NLAT+1)
C /ZP/24/ * *
TOP
SET CMSTYPE RT
```

INITIAL DISTRIBUTION LIST

		Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Research Administration, Code 012A Naval Postgraduate School Monterey, California 93943	1
3.	Professor R. E. Newton, Code 69Ne Naval Postgraduate School Monterey, California 93943	10
4.	Professor R. T. Williams, Code 63Wu Naval Postgraduate School Monterey, California 93943	5
5.	Professor A. L. Schoenstadt, Code 53Zh Naval Postgraduate School Monterey, California 93943	1
6.	Professor D. Salinas, Code 69Zc Naval Postgraduate School Monterey, California 93943	1
7.	Professor R. H. Franke, Code 53Fe Naval Postgraduate School Monterey, California 93943	1
8.	Superintendent Naval Postgraduate School Monterey, California 93943 ATTN Code 0142 Library	2
9.	Commanding Officer Naval Environmental Prediction Research Facility Monterey, California 93943	10
10.	Doctor A. N. Staniforth Recherche en Prevision Numerique Atmospheric Environment Service Dorval, Quebec H9P 1J3 CANADA	1